Computer Networks

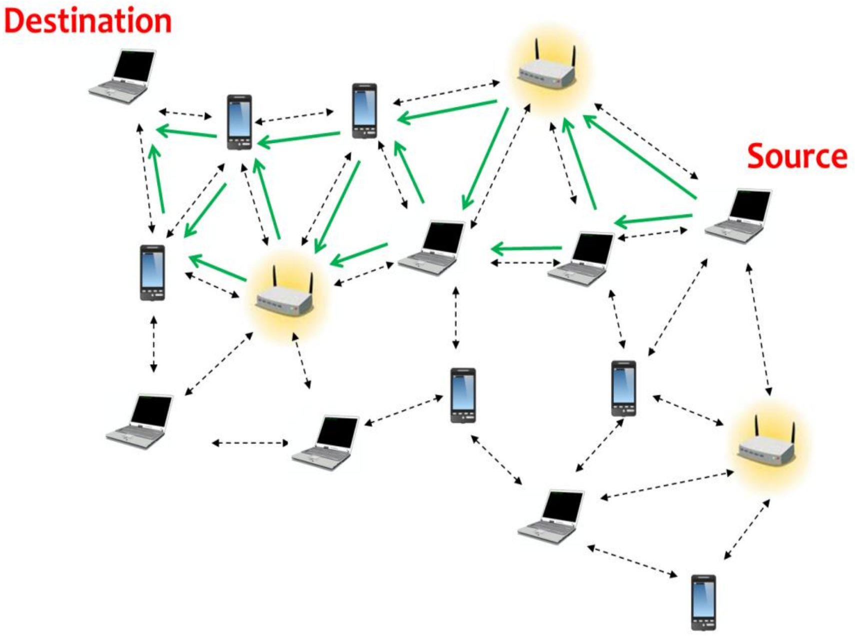
Dr. Ali Sayyed
Department of Computer Science
National University of Computer & Emerging Sciences

Network Layer

Routing Protocols

What is Routing

- Routing is the process of **path selection** in any network.
- A computer network is **made of many machines, called nodes**, and paths or links that connect those nodes.
- Communication between two nodes in an interconnected network can **take place through many different paths**.
- Routing is the process of **selecting the best path** using some predetermined rules.

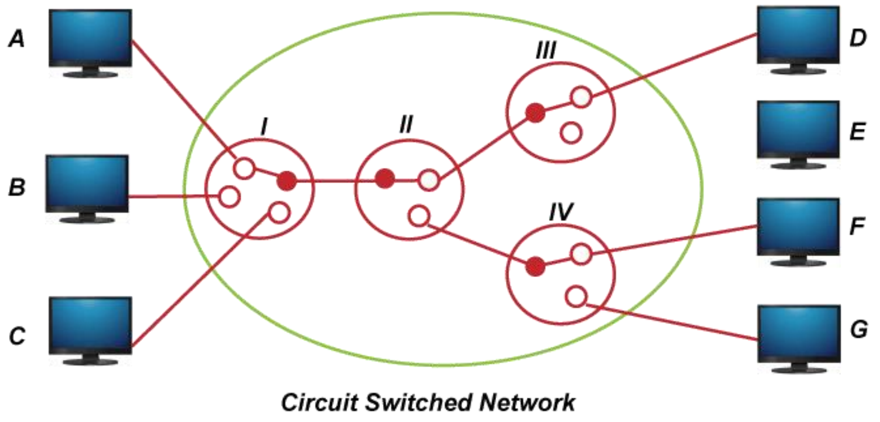


Routing Metrics and Costs

- **Hop count:** Hop count is defined as a metric that specifies the number of passes through internetworking devices.
- **Delay**: The delay values for all the links along the path is considered.
- **Bandwidth**: The capacity of the link.
- **Load**: Load refers to the degree to which the network resource such as a router or network link is busy.
- **Reliability**

Routing circuit-switched networks

- Many connections will need paths through more than one switch
- Need to find a route
  - Efficiency
  - Resilience
- A dedicated physical path, or circuit, is established and maintained between two nodes or locations for the duration of a connection
- Public telephone switches are a tree structure



Routing in Packet Switched Network

- However, **Packet Switched Network require dynamic routing** which allows changes in routes according to the network condition
- This is one of the complex and crucial aspect of packet switched networks
- Characteristics required
  - Correctness
  - Simplicity
  - Stability
  - Fairness
  - Efficiency
- Criteria for selection of route
  - Least cost in terms of hops count, delay, bandwidth etc

Routing Decision and Update Timing

- Place
  - Distributed
    - Made by each node
    - Nodes use local knowledge
  - Centralized
    - Collect info from all nodes
- Update timing
  - Fixed - never updated
  - Adaptive - regular updates

Routing Strategies

- Fixed
- Flooding
- Random
- Adaptive/Dynamic

Fixed Routing

- Single permanent route for each source to destination pair
- Determine routes using a least cost algorithm
- Route fixed, at least until a change in network topology
- Useful for
  - static networks where the network condition does not change often

CENTRAL ROUTING DIRECTORY

From Node

To Node	1	2	3	4	5	6
1	—	1	5	2	4	5
2	2	—	5	2	4	5
3	4	3	—	5	3	5
4	4	4	5	—	4	5
5	4	4	5	5	—	5
6	4	4	5	5	6	—

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Directory

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Directory

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

Node 6 Directory

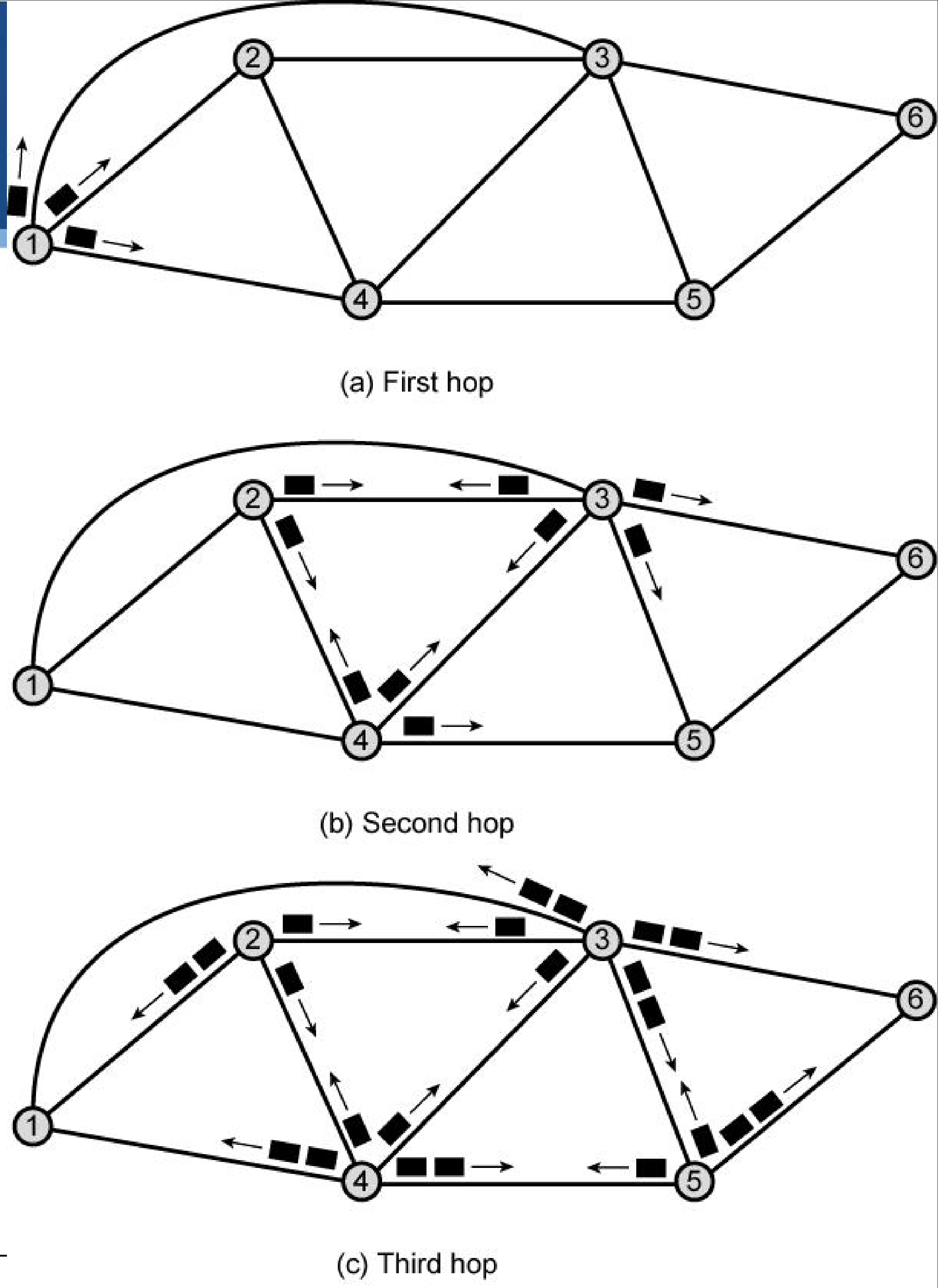
Destination	Next Node
1	5
2	5
3	5
4	5
5	5

Flooding

- No network info required
- Packet sent by node to every neighbor
- Incoming packets retransmitted on every link
- Eventually a number of copies will arrive at destination
- Each packet is uniquely numbered (sequence number) so duplicates can be discarded
- Nodes can remember packets already forwarded to keep network load in bounds

Flooding Example

- Properties of Flooding
  - All possible routes are tried
    - Very robust
  - At least one packet will have taken minimum hop count route
    - Can be used to set up virtual circuit
  - All nodes are visited
    - Useful to distribute information (e.g. routing)
- Can we use flooding for making routing tables and then selecting best path?



Random Routing

- Random routing has the simplicity and robustness of flooding with far less traffic load.
- A node select only one outgoing path to forward the incoming packet
- Selection can be random or round robin (excluding the link on which the packet arrived)
- Can select outgoing path based on probability calculation
- No network info needed
- Route is typically not least cost nor minimum hop

Adaptive Routing

- Used by most of the packet switching networks
- Routing decisions change as conditions on the network change
  - Topology
  - Failure
  - Congestion
- Requires info about network
- Decisions more complex
- Tradeoff between quality of network info and overhead
- Improved performance

Distance-vector Routing

- For each destination, each node maintain the next hop of shortest path
- Routers using distance-vector protocol **do not have knowledge of the entire path** to a destination. Instead they **maintains only neighbor states**.
  - **Direction** (Next Hop) to which a packet should be forwarded.
  - **Cost**
    - Cost to reach a certain node, calculated using various route metrics. Examples are **hop count**, **node delay**, **link quality** and available **bandwidth**.
- Each node **periodically send everything** it know to neighbors
- Slow convergence for large Networks, however good for small networks
- e.g. Bellman-Ford algorithm

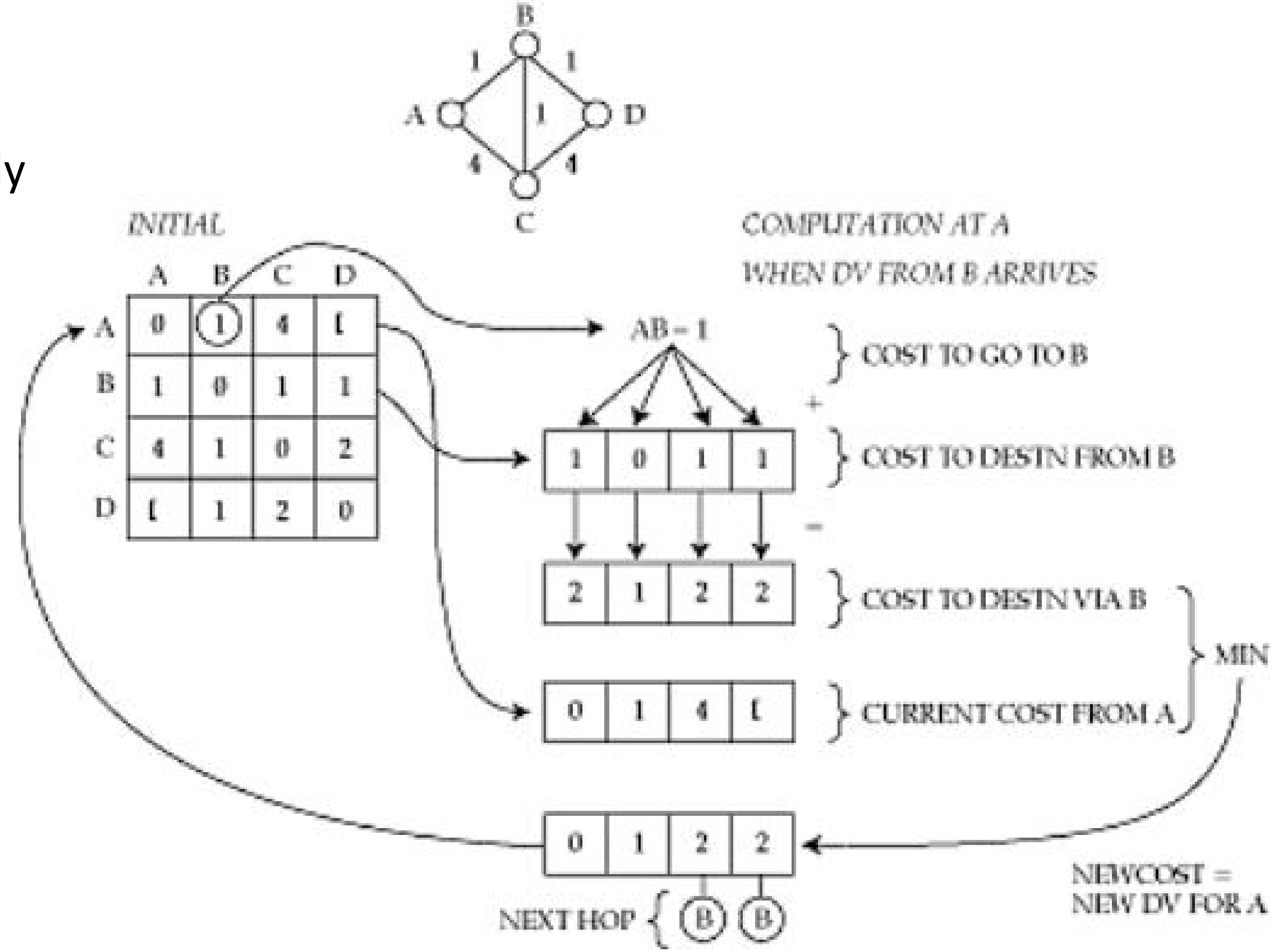
Bellman-Ford Algorithm Definitions

- The Bellman–Ford algorithm computes shortest paths from a source node to all of the other nodes in a network.

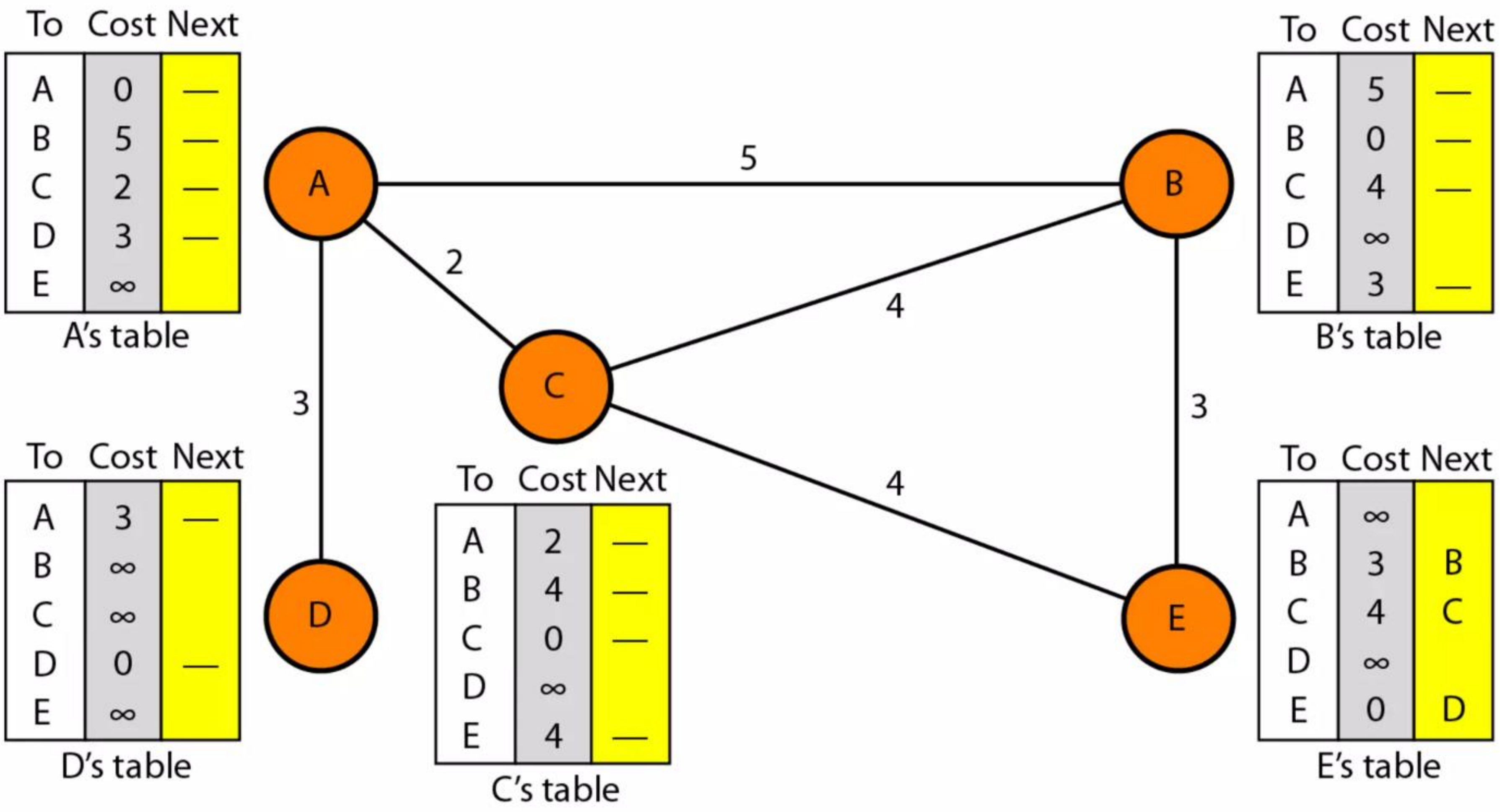
1. Each node calculates the distances between itself and all known nodes and stores this information as a table.
2. Each node sends its table to all neighboring nodes.
3. When a node receives distance tables from its neighbors, it calculates the shortest routes to all other nodes and updates its own table to reflect any changes.

Bellman-Ford Algorithm Routing Example

- w(i, j) = link cost from node i to node j
  - w(i, i) = 0
  - w(i, j) = ∞ if the two nodes are not directly connected
  - w(i, j) ≥ 0 if the two nodes are directly connected



Bellman-Ford Algorithm Routing Example



Bellman-Ford Algorithm Routing Example

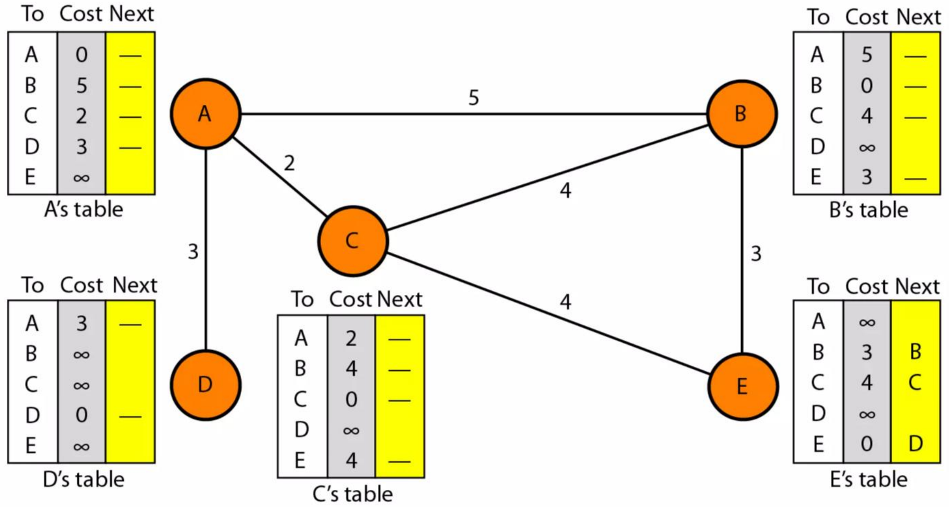
a) Idea is to share the information between neighbors.

b) The node A does not know the distance about E, but node C does.

c) If node C share it routing table with A, node A can also know how to reach node E.

d) On the other hand, node C does not know how to reach node D, but node A does.

e) If node A share its routing table with C, then node C can also know how to reach node D.



Bellman-Ford Algorithm Routing Example

To	Cost
A	2
B	4
C	0
D	∞
E	4

Received from C

+2

To	Cost	Next
A	4	C
B	6	C
C	2	C
D	∞	C
E	6	C

A's modified table

Compare

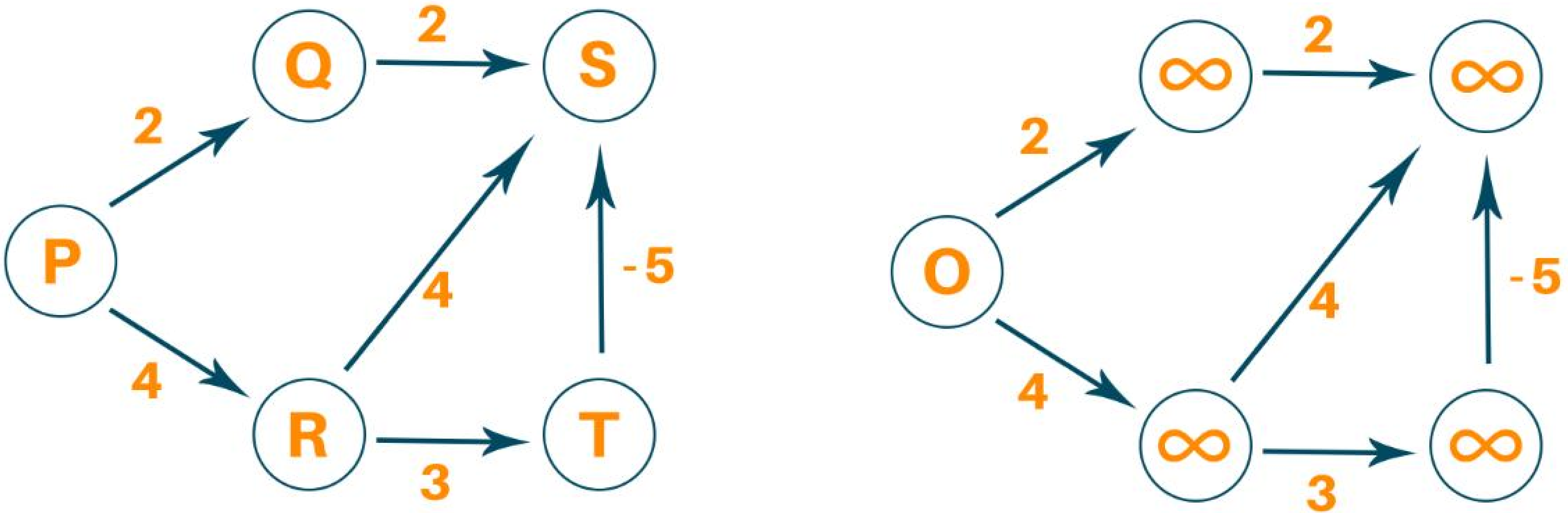
To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

A's old table

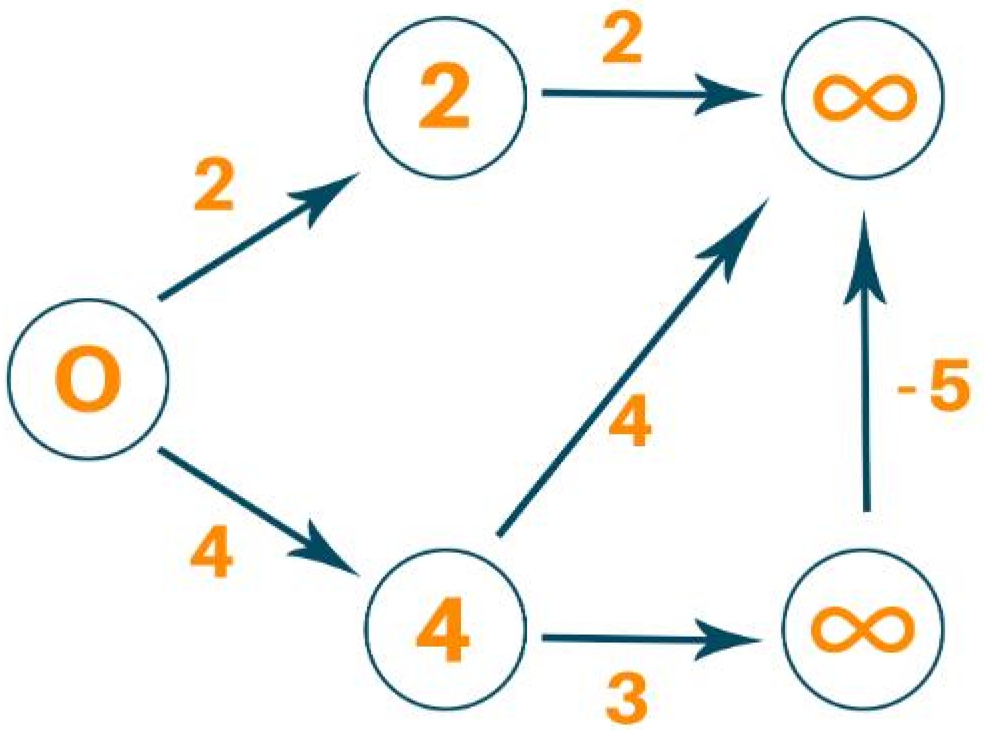
To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

A's new table

Bellman-Ford Algorithm Routing Example

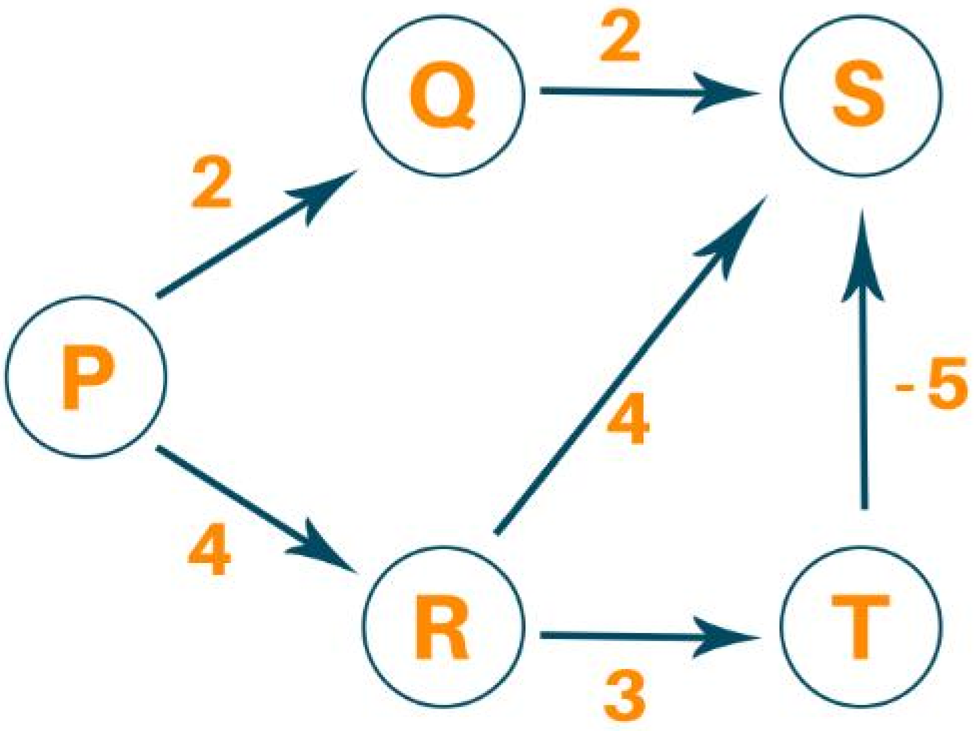


Bellman-Ford Algorithm Routing Example



Bellman-Ford Algorithm Routing Example

Bellman-Ford Algorithm Routing Example



	Q	R	S	T
O	∞	∞	∞	∞
O	2	4	∞	∞
O	2	4	4	7
O	2	4	2	7
O	2	4	2	7

Link-state routing

- Every node **maintain complete topology** database in the form of a map of connectivity to the network
- The map shows which nodes are connected to which other nodes.
- Each node then independently calculates the best path to every possible destination in the network.
- Every node inform all the nodes in a network of topology changes
- send info about your neighbors to everyone
- Each collection of best paths will then form each node's routing table.
- Better than Distance Vector for large Networks
- E.g. Dijkstra's algorithm

Link-state routing working

- Each router keeps track of its incident links and **create a link state table**
  - Whether the link is up or down
  - The cost on the link
- Each router broadcasts the link state table (**Flooding**)
  - To give every router a complete view of the graph
- Each router runs **Dijkstra's algorithm**
  - **To compute the shortest paths**
  - … and construct the forwarding table

When to Flood

- **Triggered by a topology change**
  - Link or node failure/recovery or
  - Configuration change like updated link metric
  - Converges quickly, but can cause flood of updates
- **Periodically**
  - Typically (say) every 30 minutes
  - Corrects for possible corruption of the links
  - Limits the rate of updates, but also failure recovery

Dijkstra's Algorithm Definitions

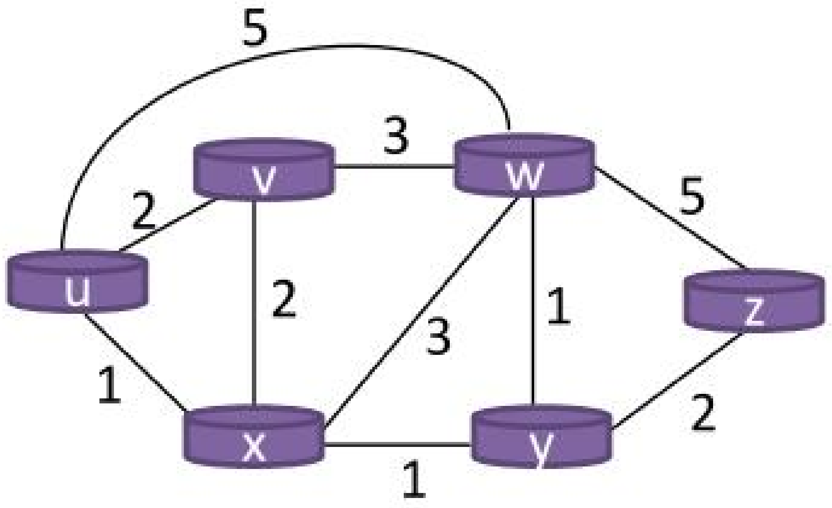
- Iterative algorithm which find shortest paths from given source node to all nodes
- D(v): Cost from source node s to destination node v
- p(v): previous node along the least-cost path from source.
- N: set of nodes to which the least-cost path is found.
- s = source node
- w(i, j) = link cost from node i to node j
- L(n) = cost of least-cost path from node s to node n currently known
  - At termination, L(n) is cost of least-cost path from s to n

Dijkstra’s Algorithm Method

- Step 1 [Initialization]
  - N = {s} Set of nodes so far incorporated consists of only source node
- Step 2 [Get Next Node]
  - Find neighboring node not in N with least-cost path from s
  - Incorporate node into N
- Step 3 [Update Least-Cost Paths]
  - $L(n) = \min[L(n), L(x) + w(x, n)]$
  - If latter term is minimum, **path from s to n** is path from **s to x** plus **x to n**
- Algorithm terminates when all nodes have been added to N

Dijkstra’s algorithm: Example

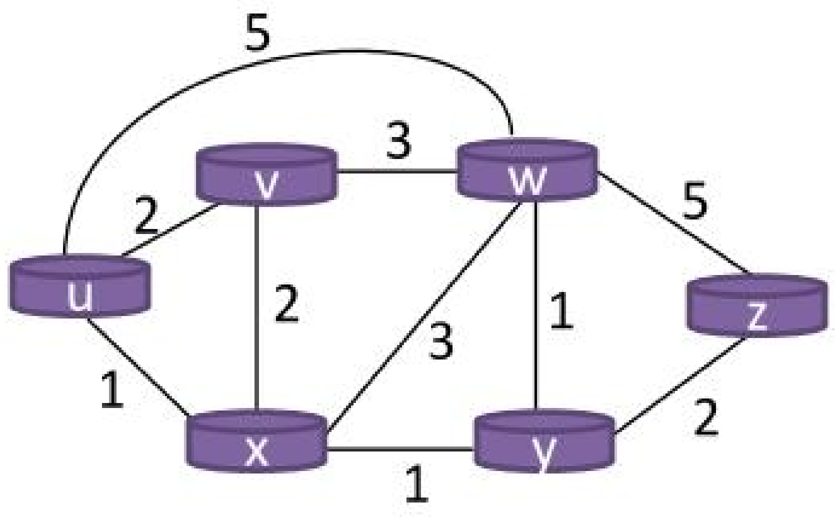
Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞

Dijkstra's algorithm: Example

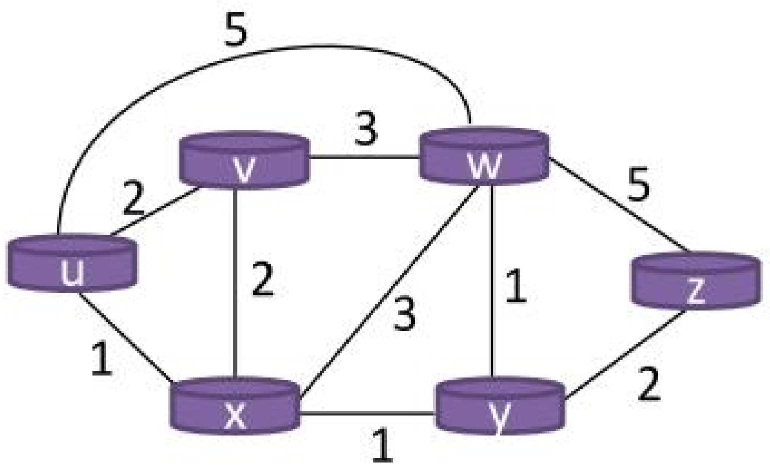
Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞

Dijkstra's algorithm: Example

Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Link-State Routing - Advantages

- **Fast Network Convergence**: It is the main advantage of the link-state routing protocol. Because of receiving an LSP, link-state routing protocols immediately flood the LSP out.
- **Topological Map:** Link-state routing uses a topological map for creating the network topology. Using the map, each router can separately determine the shortest path to every network.
- **Event-driven Updates:** After initial flooding of LSPs, the LSPs are sent only when there is a change in the topology and contain only the information regarding that change. The LSP contains only the information about the affected link. The link-state never sends periodic updates.

Link-State Routing - Disadvantages

- **Memory Requirements** – The link-state routing protocol creates and maintains a database. The database required more memory than a distance vector protocol.
- **Processing Requirements** – Link-state routing protocols also require more CPU processing because the algorithm requires more CPU time than distance-vector algorithms just like Bellman-Ford because link-state protocols build a complete map of the topology.
- **Bandwidth Requirements** – The link-state routing protocol floods link-state packet during initial start-up and also at the event like network breakdown, and network topology changes, which affect the available bandwidth on a network. If the network is not stable it also creates issues on the bandwidth of the network.